Python: module vcs.gui_template_editor

vcs.gui_template_editor

index

```
# Template Editor GUI Module
```

Modules

vcs.Canvas	cdms	browser.gui_message	sys
MA	vcs.fonteditorgui	gui_support	tkFileDialog
Numeric	browser.gui_busy	vcs.lineeditorgui	vcs
Tkinter	gui_support.gui_color	os	
vcs._vcs	browser.gui_control	string	

Classes

Template_Editor
busy
create_attribute
template_state

class ***Template_Editor***

Methods defined here:

__init__(self, gui_parent=None, canvas=None, plot=None, template_name='quick', template_orig_name='quick', graphics_method='isofill', graphics_method_name='default')

copy_template(self)

create_axis(self, page)

create_borders_and_lines(self, page)

create_data_and_legend(self, page)

create_labels(self, page)

```
        #--------------------------------------------------
        # Populate the 'Labels' tab
        #--------------------------------------------------
```

create_menu_bar(self)

create_new(self)

do_nothing(self, event)

```
#-----------------------------------------------------------
# Don't do anything.
#-----------------------------------------------------------
```

evt_change_color(self, master, obj, event)

```
#-----------------------------------------------------------
# Change the color of the text window when inputing text.
#-----------------------------------------------------------
```

evt_create_new(self, result)

execute(self, result, template_name=None)

```
# Main command control function.  Figure out what to do with
#  request (ie. Save, Cancel, etc.)
```

execute2(self, name, result)

```
#-----------------------------------------------------------
# Handle the closing of the "Properties" button
#-----------------------------------------------------------
```

exit(self)

get_template(self)

```
#-----------------------------------------------------------
# Create a copy of the template and then tell the canvas to u
#  the template copy to display the plot. (This way we're not
#  actually modifying the real template in case we want to
#  revert or do a "Save As".)
#-----------------------------------------------------------
```

open_new(self)

priority_change(self, name, evt=None)

refresh_data(self)

refresh_toggle(self, attr_name, toggle_val)

reinitialize_editor(self)

remove_temp_templates(self)

replot(self)

restore_plots_on_canvas(self)

sample_data(self)

save(self, template_name=None)

```
# Prompt user to save changes if they are trying to close the
#  editor and changes have been made.
```

saveas(self)

savefile(self)

savescript(self)

savesession(self)

```
      # Verify template being saved and give users one more chance
      #  cancel their save request.
```

scale(self)

select_all(self)

set_value(self, attribute, result)

```
      #-------------------------------------------------------------
      # Change the value of a particular attribute in the
      #  template so that it reflects the state of the GUI.
      #  This is more for the pull down menu options such as
      #  text orientation, text table, etc.
      #-------------------------------------------------------------
```

show_canvas(self)

show_properties(self, parent, master)

```
      #-------------------------------------------------------------
      # Open properties button for selected attribute
      #-------------------------------------------------------------
```

unselect_all(self)

update_value(self, item, name, evt=None)

```
      #-------------------------------------------------------------
      # Update the new value for the template in use.  This is norm
      #  from a bound object when someone changes focus to another
      #  hits 'Enter'.
      #-------------------------------------------------------------
```

class ***busy***

```
    #-------------------------------------------------------------
    # Create a Dummy class
    #-------------------------------------------------------------
```

class ***create_attribute***

```
    #-------------------------------------------------------------
    # This creates a template object and packs it into the current
```

```
    #  master widget.
    #--------------------------------------------------------
```

Methods defined here:

__init__(self, name=None, parent=None, master=None, x1=None, x2=None, y1=None, y2=None,

class ***template_state***

```
    # Dictionary of checkbutton values.  This is for use
    #   in keeping track of which values are turned on and off
```

Methods defined here:

__init__(self)

Functions

change_field_mode(self, mode)

create(gui_parent=None, canvas=None, plot=None, template_name='', template_orig_name='')

```
    #--------------------------------------------------------------
    # Call mainloop() if called from the command line
    #--------------------------------------------------------------
    # Create/Popup template editor for VCS.
```

format_number(value, digits=4)

```
    #--------------------------------------------------------------
    # Format printed number so it is only has 'x' digits after the
    #  decimal
    #--------------------------------------------------------------
```

spacer(interior, height=1, width=1)

```
    #--------------------------------------------------------------
    # Create an invisible spacer to get around padx and pady
    #  putting spaces on both sides
    #--------------------------------------------------------------
```

toggle_on_off(self, parent)

```
    #--------------------------------------------------------------
    # When an attribute is selected to be displayed, highlight the
    #  label name, activate the coordinate entryfields and activate
    #  the properties button. When it is unselected, do the reverse.
    #--------------------------------------------------------------
```

Data

Pmw = <Pmw.Pmw_1_2.lib.PmwLoader.PmwLoader instance>
hot_color = 'green'